**Assignment - Full Stack Developer.**

**Task Management Application Documentation**

KOUSHIK G | koushikgdatta5@gmail.com

1. Introduction:

The Task Management Application is a web-based platform designed to help users organize and track their tasks effectively. With features for creating, editing, and deleting tasks, as well as viewing detailed information, the application aims to streamline task management processes for individuals or teams.

1. Front-end Implementation:

* User Interface Components:

- Landing page displaying a list of tasks.

- Form for adding new tasks with title, description, and due date.

- Detailed view for each task.

- Edit and delete options for existing tasks.

* Design Choices:

- Utilization of HTML, CSS, and JavaScript for front-end development.

- Responsive design to ensure usability on both desktop and mobile devices.

- Minimalistic yet intuitive interface for improved user experience.

* Challenges Faced:

- Ensuring cross-browser compatibility and consistent layout across different devices.

- Implementing dynamic elements such as edit and delete functionalities without compromising performance.

- Incorporating user feedback for iterative design improvements.

* Future Improvements:

- Enhancing visual design with modern UI frameworks or libraries.

- Implementing advanced features like drag-and-drop task organization.

- Integrating real-time collaboration for team-based task management.

3. Back-end Implementation:

* RESTful API:

- Handles CRUD operations for tasks.

- Endpoints for retrieving all tasks, creating new tasks, updating existing tasks, and deleting tasks.

* Server-side Technology:

- Utilization of Node.js with Express framework for back-end development.

- PostgreSQL database for storing task data.

* Challenges Faced:

- Ensuring data consistency and integrity during CRUD operations.

- Managing database connections and optimizing query performance.

- Implementing error handling and validation for robust API functionality.

* Future Enhancements:

- Implementing authentication and authorization mechanisms for user accounts.

- Integrating data analytics for task performance insights.

- Scaling the application to support larger user bases and increased data volume.

1. Additional Features and Improvements:

* Suggestions for additional features:

- Task categorization and filtering options.

- Reminders and notifications for upcoming tasks.

- Integration with external calendars or productivity tools.

* Security measures:

- Encryption of sensitive user data to ensure privacy and confidentiality.

- Regular security audits and updates to address potential vulnerabilities.

* Performance optimization:

- Caching mechanisms to reduce database load and improve response times.

- Minification and bundling of front-end assets for faster page loading.

* User feedback and iterative improvements:

- Establishing channels for users to provide feedback and suggestions.

- Iterating on features based on usability testing and user behavior analysis.

5. Conclusion:

The Task Management Application provides a robust solution for organizing tasks and improving productivity. With a user-friendly interface, efficient backend functionality, and room for future enhancements, the application is poised to meet the evolving needs of users and adapt to changing requirements over time.

6. Appendix:

- Code snippets for key components of the application.

**Index.js :**// Import required modules

import express from "express";

import bodyParser from "body-parser";

import pg from "pg";

// Initialize Express app and database connection

const app = express();

const port = 3000;

const db = new pg.Client({

  user: "postgres",

  host: "localhost",

  database: "PERMALIST",

  password: "admin@123",

  port: 5432,

});

db.connect();

// Middleware setup

app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static("public"));

// Routes

app.get("/", async (req, res) => {

  try {

    const result = await db.query("SELECT \* FROM items;");

    const items = result.rows;

    res.render("index.ejs", {

      listTitle: "Today",

      listItems: items,

    });

  } catch (err) {

    console.error("Error fetching tasks:", err);

    res.status(500).send("Error fetching tasks");

  }

});

app.post("/add", async (req, res) => {

  const item = req.body.newItem;

  try {

    // Perform input validation

    if (!item) {

      return res.status(400)

        .send(`<p style="font-family: 'Indie Flower', cursive; font-weight: 300px ; font-size: 20px">Please enter the "TASK"</p>

        <h3><a href="/" style="font-family: 'Indie Flower', cursive;">Go back</a></h3>`);

    }

    await db.query("INSERT INTO items (title) VALUES ($1)", [item]);

    res.redirect("/");

  } catch (err) {

    console.error("Error adding task:", err);

    res.status(500).send("Error adding task");

  }

});

app.post("/edit", async (req, res) => {

  const item = req.body.updatedItemTitle;

  const id = req.body.updatedItemId;

  try {

    // Perform input validation

    if (!item || !id) {

      return res.status(400)

        .send(`<h1 style="font-family: 'Indie Flower', cursive; font-weight: 300px ; font-size: 20px">Shouldn't leave empty while editing the "TASK"</h1>

      <h3><a href="/" style="font-family: 'Indie Flower', cursive; ">Go back</a></h3>`);

    }

    await db.query("UPDATE items SET title = ($1) WHERE id = $2", [item, id]);

    res.redirect("/");

  } catch (err) {

    console.error("Error updating task:", err);

    res.status(500).send("Error updating task");

  }

});

app.post("/delete", async (req, res) => {

  const id = req.body.deleteItemId;

  try {

    // Perform input validation

    if (!id) {

      return res.status(400).send("<h1>Task ID is required.</h1>");

    }

    await db.query("DELETE FROM items WHERE id = $1", [id]);

    res.redirect("/");

  } catch (err) {

    console.error("Error deleting task:", err);

    res.status(500).send("Error deleting task");

  }

});

// Start the server

app.listen(port, () => {

  console.log(`Server running on port ${port}`);

});

**Index.ejs:**<%- include('partials/header.ejs'); -%>

  <div class="box" id="heading">

    <h1>

      <%= listTitle %>

    </h1>

  </div>

  <div class="box">

    <% for(let item of listItems){%>

      <div class="item">

        <form action="/delete" method="post">

          <input type="checkbox" onchange="this.form.submit()" name="deleteItemId" value="<%= item.id %>">

        </form>

        <p id="title<%=item.id%>">

          <%= item.title %>

        </p>

        <form class="edit" action="/edit" method="post">

          <input type="hidden" name="updatedItemId" value="<%= item.id %>">

          <input id="input<%=item.id%>" type="text" name="updatedItemTitle" value="<%= item.title %>" autocomplete="off"

            autofocus="true" hidden="true" />

          <button id="done<%=item.id%>" class="edit" type="submit" hidden><img class="icon"

              src="/assets/icons/check-solid.svg" alt="tick image"></button>

        </form>

        <!-- Edit button -->

        <button id="edit<%=item.id%>" class="edit" onclick="handler('<%=item.id%>')"><img class="icon"

            src="/assets/icons/pencil-solid.svg" alt="pencil image"></button>

      </div>

      <% } %>

        <form class="item" action="/add" method="post">

          <input type="text" id="white" name="newItem" placeholder="  Add  'TASK'" autocomplete="off"

            autofocus="true" />

          <button class="add" type="submit" name="list" value=<%=listTitle %> >+</button>

        </form>

  </div>

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

  <script>

    function handler(id) {

      $("#edit" + id).hide();

      $("#title" + id).hide();

      $("#done" + id).show();

      $("#input" + id).show().focus();

    }

  </script>

  <%- include('partials/footer.ejs'); -%>

**Queries.sql:**CREATE TABLE items (

  id SERIAL PRIMARY KEY,

  title VARCHAR(100) NOT NULL

);

INSERT INTO items (title) VALUES ('Buy milk'), ('Finish homework');

- Screenshots showcasing the user interface design.



